CSE [8, 8, 8, 8]B Today

PSA 1: `fourInARow`
Due Monday 11:59pm – TUTOR HOURS START TOMORROW!

```
return, arrays and methods
```

```
String and char
```
PSAs: rules for working with a partner

[I] PSA1: INDIVIDUAL → IT WILL BE MARKED with [I] on the web

PSA2: YOU CHOOSE

• Please do!
(1000s of students, only 100s of lab machines—you do the math)

• You must PAIR PROGRAM
  – You must always be together when coding
  – One person drives, one person navigates

• You must include in your header comment:
  – Both partners’ names
  – Both partners’ PIDs
  – Both partners’ cs8b logins
  (cs8sb... look for it following instructions in PSA1)
1. Is a Java String an Object or a primitive type? Is a Java char an Object or a primitive type?

A. String is an Object, char is an Object

B. String is a primitive, char is an Object

C. String is a primitive, char is a primitive

D. String is an Object, char is a primitive
2. What Java statement will test whether two variables of type String, s1 and s2, store the same text?

A. s1.sameAs( s2 )

B. s1.equals( s2 )

C. s1 == s2

D. s1 && s2
2. You have a variable named `myString` that stores the String "CSE8B Rocks!". What Java statement will return the substring "Rocks" from `myString`?

A. `myString.substring( 0 )`

B. `myString.substring( 6 )`

C. `myString.substring( 6, 11 )`

D. `myString.concat( "Rocks" )`
4. You have created a variable named `myCharacter` as follows: 
   ```java
   Character myCharacter = new Character('M');
   ```
   The value of `myCharacter.compareTo('N')` is -1. Explain why this is true.

   A. -1 is returned for any letter that comes after M in the alphabet

   **B.** N’s Unicode (ASCII) code is one greater than M’s

   C. -1 is an error code returned by this method
The following CODE is in a class called ArrayPlay. ALMOST identical to the class you will work with in PSA1 (Except it has a different name so that I can provide you with the ArrayPlay code separately from the assignment starter code. But the examples should be very similar to your assignment.)
CLASS
The ArrayPlay class

/** ArrayPlay is a class that implements some basic matching functions on a list */
public class ArrayPlay
{
    private int[] storedArray; // FIELD!

    public ArrayPlay( int[] inputArray ) // CONSTRUCTOR!!
    {
        storedArray = new int[inputArray.length];
        for ( int i = 0; i < inputArray.length; i++ )
        {
            storedArray[i] = inputArray[i];
        }
    }

    public static void main( String[] args )
    {
        int[] a = {1, 2, 3};
        ArrayPlay am = new ArrayPlay( a );
        a[0] = 42;
        System.out.println( am.storedArray[0] );
    }
}

What does the last line of main print?  A. 0  B. 1  C. 42  D. Other
METHODS
Methods: the Heart of Computation

Javadoc comments! Use them!

/** greaterThanN returns true if the element at position index in storedArray is greater than num.
 * @param num The N to compare against
 * @param index The index of the element in question
 * @return true if storedArray[index] is greater than num, false otherwise
 */
public ____A____ greaterThanN( ____B____ index, int num )
{
    if ( ______C_________________ )
        return true;
    else
        return false;
}

What should go in blank (A)?
A. void
B. int
C. boolean
D. int[]
E. Other
Methods: the Heart of Computation

Javadoc comments! Use them!

/**
greaterThanN returns true if the element at position index in
* storedArray is greater than num.
*
* @param num The N to compare against
* @param index The index of the element in question
* @return true if storedArray[index] is greater than num, false otherwise
* */

public boolean greaterThanN( ___(B)___ index, int num )
{
    if ( _______(C)_______________ )
        return true;
    else
        return false;
}
Methods: the Heart of Computation

Javadoc comments! Use them!

/** greaterThanN returns true if the element at position index in storedArray is greater than num.
* @param num The N to compare against
* @param index The index of the element in question
* @return true if storedArray[index] is greater than num, false otherwise
* */

    public boolean greaterThanN( int index, int num )
    {
        if ( ______(C)_______________ )
            return true;
        else
            return false;
    }

What should go in blank (C)?
A. index > num
B. num > index
C. storedArray[index] > num
D. this.storedArray > num
E. this.index > num
Methods: the Heart of Computation

Javadoc comments! Use them!

/** greaterThanN returns true if the element at position index in
* storedArray is greater than num.
* *
* @param num The N to compare against
* @param index The index of the element in question
* @return true if storedArray[index] is greater than num, false otherwise
* */

public boolean greaterThanN( int index, int num )
{
  if ( storedArray[index] > num )
    return true;
  else
    return false;
}

Mmm.... Some problems??

What/where is storedArray? → it’s an instance variable (this.storedArray)

How do we call this method?
public class ArrayPlay

    // FIELDS
    private int[] storedArray;
    // CONSTRUCTOR not shown
    // METHODS
    public boolean greaterThanN(int index, int num)
    {
        if (myArray[index] > num)
            return true;
        else
            return false;
    }
    public static void main(String[] args)
    {
        int[] myA = {2, 4, 10, 1};
        new ArrayPlay(myA);
        System.out.println(ap.greaterThanN(2, 6));
    }

    • All methods must be defined in a class
    • To call them method, you must have an object
      (unless the method is static)
    • There is a special method (main) that gets called automatically

Don't forget the filename has to be ArrayPlay.java
Java: It’s about classes (and objects)

```java
public class ArrayPlay {
    private int[] storedArray;
    // Constructor not shown
    public boolean greaterThanN(int index, int num) {
        if (storedArray[index] > num)
            return true;
        else
            return false;
    }

    public static void main(String[] args) {
        int[] myA = {2, 4, 10, 1};
        ArrayPlay ap = new ArrayPlay(myA);
        System.out.println(ap.greaterThanN(2, 6));
    }
}
```

What will this method print?

A. true
B. false
C. It will cause an error
D. I don’t know
public class ArrayPlay
private int[] storedArray;
// Constructor not shown
public boolean greaterThanN( int index, int num )
{
    if ( storedArray[index] > num )
        return true;
    else
        return false;
}
public static void main( String[] args )
{
    int[] myA = {2, 4, 10, 1};
    ArrayPlay ap = new ArrayPlay( myA );
    System.out.println( ap.greaterThanN( 6, 2 ) );
}

What will this method print?
A. true
B. false
C. It will cause an error
D. I don’t know

Out of Bounds!
LOOPS
Getting loopy... worksheet!

Complete the following method which returns true if any element in myArray is greater than num, and false if no element is greater than num.

```java
public boolean anyGreaterThanN( int num )
{
    // code goes here
}
```

Which of the following correctly completes the method?

A. ```java
for ( int x : storedArray )
{
    if ( x > N ) {
        return true;
    } // end if
} // end for
else {
    return false;
} // end else
} // end for
```

B. ```java
for ( int x : storedArray )
{
    if ( x > N ) {
        return true;
    } // end if
} // end for
```

C. ```java
for ( int x : storedArray )
{
    if ( x > N ) {
        return true;
    } // end if
} // end for
return false;
```
PARAMETERS and RETURN
```java
public boolean anyGreaterThanN(int num) {
    for (int x : storedArray) {
        if (x > num) {
            return true;
            System.out.println(“Found it!”);
        }
    }
    return false;
}
```

```java
public static void main(String[] args) {
    int[] myA = {2, 4, 6, 1, 15};
    ArrayPlay ap = new ArrayPlay(myA);
    ap.anyGreaterThanN(1);
}
```

What will be printed?
A. Nothing
B. “Found it!”
C. “true”
D. “true” “Found it!”

Once we run “return” → THE METHOD ENDS RIGHT THERE!
Actually the compiler will complain...
public boolean match( int[] toMatch )
{
    for ( int i = 0; i < storedArray.length; i++ )
    {
        if ( storedArray[i] != toMatch[i] )
        {
            toMatch[i] = storedArray[i];
        }
    }
    return true;
}

public static void main( String[] args )
{
    int[] a1 = {2, 4, 6, 1, 15};
    ArrayPlay ap = new ArrayPlay(a1);
    int[] myA = { 1, 1, 1, 1, 1 };
    ap.match( myA );
}

What is the value of myA at the end of main?
A. [2, 4, 6, 1, 15]  B. [1, 1, 1, 1, 1]  D. true  C. Other
public boolean match( int[] toMatch )
{
    for ( int i = 0; i < storedArray.length; i++ )
    {
        if ( storedArray[i] != toMatch[i] )
        {
            toMatch[i] = storedArray[i];
        }
    }
    return true;
}

public static void main( String[] args )
{
    int[] a1 = {2, 4, 6, 1, 15};
    ArrayPlay ap = new ArrayPlay(a1);
    int[] myA = {1, 1, 1, 1, 1};
    ap.match(myA);
}

1. As you reach each variable, make a box for it
2. If the variable holds a primitive, put the value in the box
3. If the variable holds an object (or an array) put an arrow in the box
4. Parameters are always passed BY VALUE
   (by COPYING WHAT’S IN THE BOX, the arrow or the value)
Summary so far ... 

- Computation takes place (generally) by calling methods
- Methods can return values and modify the data passed to them
- Parameters are passed by value to methods
- Primitives and objects/arrays are represented differently in memory (box-and-arrows)
- There are often many ways to solve the same problem.

Tips for success on PSA1

- Compile and run A LOT
- Work in the lab
- Trace your code on paper (no “debugging by random perturbation”)
- Think through your approach before you code
- Ask for help, don’t waste your time if you are really stuck.

(if you get really stuck with PSA1, I’ll go to TAs or instructor office hours ASAP!)

PSA1 has to be done INDIVIDUALLY! (other PSAs you’ll choose)
Strings and char and Character class
PSA2 – processing strings

“Strings” and ‘c’ ‘h’ ‘a’ ‘r’ ‘s’ in Java

Implementing your very own Secret Decoder Ring in Java...

... and a little Scrabble to warm up

PSA2 Will be posted TUESDAY.
START EARLY! It’s MUCH harder than PSA1
Chars and ints: deceptively similar

(1) > (int)'A'
(2) > 'A' + 2
(3) > (char)('A' + 2)

Can I do this????
Chars and ints: deceptively similar

What is the value of each of the expressions below:

(1) > (int)'A'
(2) > 'A' + 2
(3) > (char)('A' + 2)

Remember, casting a value changes its type

A. error, error, error
B. 65, error, error
C. 65, ‘C’, ‘C’
D. 65, 67, ‘C’
Chars and ints: deceptively similar

What is the value of each of the expressions below:

\[
\begin{align*}
(1) & \quad > \quad (\text{int})'A' \\
(2) & \quad > \quad 'A' + 2 \\
(3) & \quad > \quad (\text{char})('A' + 2)
\end{align*}
\]

but apparently types change even without a cast

Important: when you add an int to a char, it automatically becomes an int.
To get a char you have to cast it back to a char.
public class StringPlay {
    public boolean hasLetter(String word, char letter) {
        for (int i = 0; i < word.length(); i++) {
            if (word.charAt(i) == letter) {
                return true;
                System.out.println("Found it!");
            }
        }
        return false;
    }
}

> StringPlay s = new StringPlay();
> System.out.println(s.hasLetter("Sleep", 'S'));

1 error found:
File: ... StringPlay.java  [line: ... ]
Error: ... StringPlay.java: ... : unreachable statement
public class StringPlay {
    public boolean hasLetter(String word, char letter) {
        for (int i = 0; i < word.length(); i++) {
            if (word.charAt(i) == letter) {
                return true;
            }
        }
        return false;
    }
}

StringPlay s = new StringPlay();
System.out.println(s.hasLetter("Sleep", 's'));
chars Big and Small

```java
public class StringPlay {
    public boolean hasLetter( String word, char letter )
    {
        for ( int i = 0; i < word.length(); i++ )
        {
            if ( word.charAt( i ) == letter )
            {
                return true;
            }
        }
        return false;
    }

    StringPlay s = new StringPlay();
    System.out.println( s.hasLetter( "Sleep", 's' ) );
}
```

If we want this to return true, how could we ignore case?
The **Character** class *(very useful—check your book)*

```java
public class StringPlay {
    public boolean hasLetter( String word, char letter ) {
        letter = Character.toLowerCase( letter );
        for ( int i = 0; i < word.length(); i++ ) {
            char current = Character.toLowerCase( word.charAt( i ) );
            if ( current == letter ) {
                return true;
            }
        }
        return false;
    }
}
```
public class StringPlay {
    public boolean hasLetter(String word, char letter) {
        letter = Character.toLowerCase(letter);
        for (int i = 0; i < word.length(); i++) {
            char current = Character.toLowerCase(word.charAt(i));
            if (current == letter) {
                return true;
            }
        }
        return false;
    }
}

StringPlay s = new StringPlay();
String check = new String("Sleep");
s.hasLetter(check, 's');
System.out.println(check);
References, primitives, assignment
public class StringPlay {
    public boolean hasLetter( String word, char letter ) {
        letter = Character.toLowerCase( letter );
        for ( int i = 0; i < word.length(); i++ ) {
            char current = Character.toLowerCase( word.charAt( i ) );
            if ( current == letter ) {
                return true;
            }
        }
        return false;
    }
    
    StringPlay s = new StringPlay();
    String check = new String( "Sleep" );
    char let = 's';
    s.hasLetter( check, let );
    System.out.println( check );
}

In fact, strings are immutable: If you WANT to change a string, you must make a new string.
Draw the memory model for this solution

Complete the following method to replace all instances of one character with another in a string (and the return the resulting string). Your method should be case sensitive.

e.g., replaceChar( “One for the money, YO!”, 'O', 'i' ) → “ine for the money, Yi!”

public String replaceChar( String s, char gone, char here )
{
    char[] chars = s.toCharArray();
    for ( int i = 0; i < chars.length; i++ )
    {
        if ( chars[i] == gone )
            chars[i] = here;
    }

    return String.valueOf( chars );
}

Extra: Can you replace both lower and upper case instances and preserve case?